Фреймворки логгирования: логгирование в Java

Рассматриваемые вопросы

1. Логгирование и требования к нему
2. Обзор основных фреймворков логгирования
3. Использование Log4j и SLF4j

Логгирование и требования к нему

Логи (лог-файлы) — это файлы, содержащие системную информацию работы сервера или компьютера, в которые заносятся определенные действия пользователя или программы

Логгирование и требования к нему

- Разбор ошибок
- Наблюдение за работой программы
- Дебаг
- Контроль работы пользователей
- Статистика

Логгирование и требования к нему

Ошибки необходимо логировать в том классе, где они возникают. Даже в случае проброса ошибки дальше.

Необходимо логировать все:

- ход исполнения программы,
- ошибки,
- статистику

Стоимость логгирования

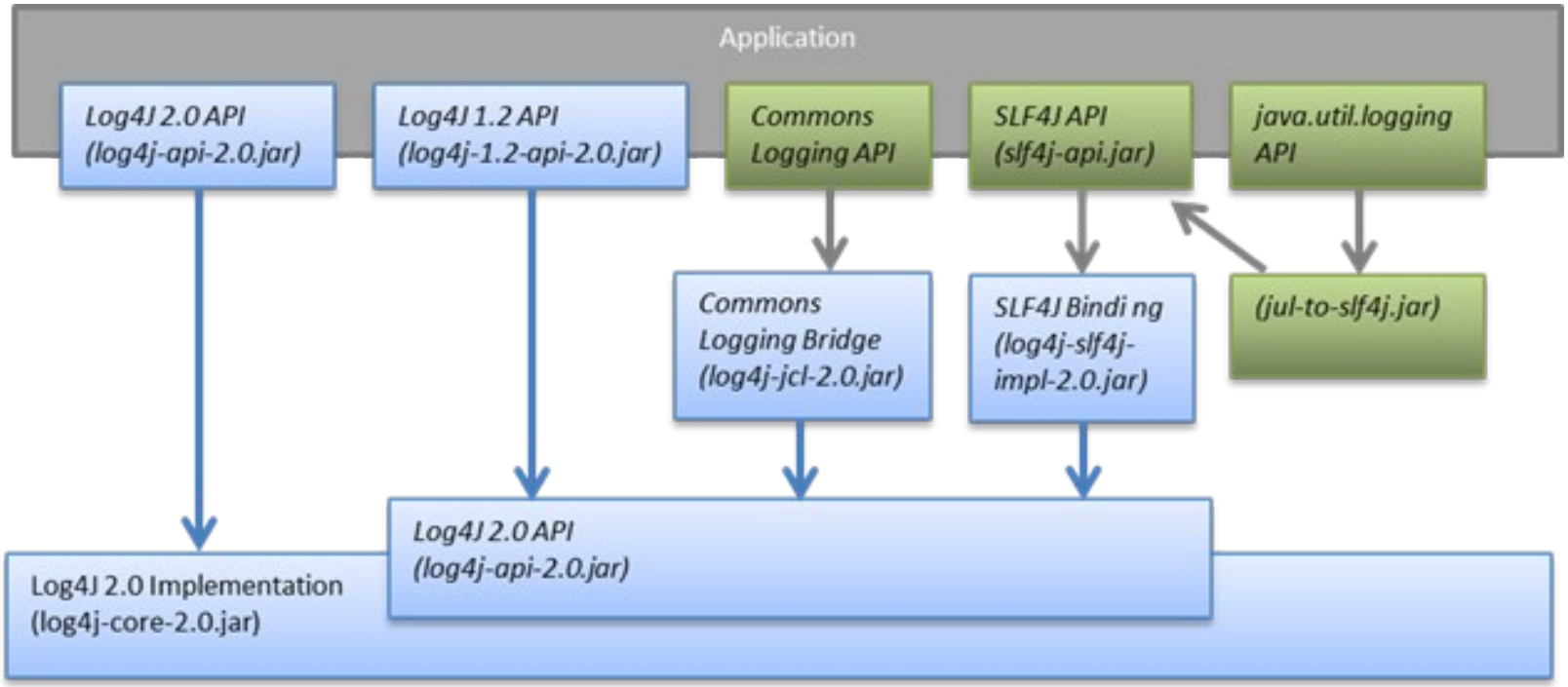
Следовательно, логов должно быть столько, сколько нам надо. Не больше, но и не меньше. Информация в логе:

- Дата
- Время
- Класс
- Пакет (модуль)
- Метод
- Сообщение

Составляющие лог-фреймворков

- Layout – отвечает за формат сообщения. Он формирует само сообщение. Например, текст, XML, или JSON. Какая информация туда попадет. Отвечает на вопрос «Что»
- Appender – отвечает за вывод сообщения. То есть, за место назначения сообщения. Например, файл, бд, консоль, веб-сервер. Отвечает на вопросы «Куда» и «Как».

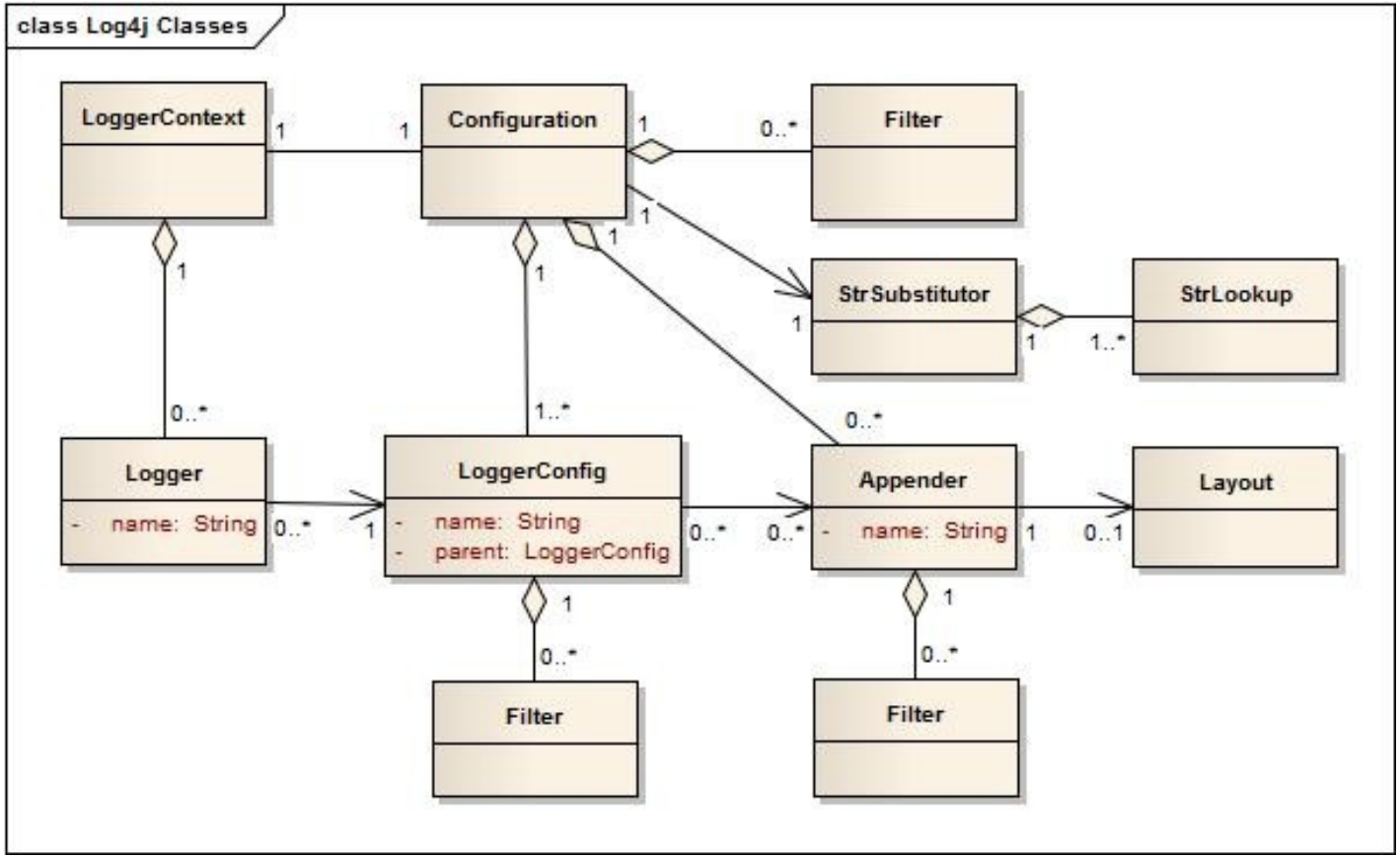
Фреймворки логгирования



Log4J

Log4J на данном этапе де-факто является промышленным стандартом и другие фреймворки пытаются копировать его интерфейс

Log4J


class Log4j Classes
LoggerContext
Configuration
Filter
StrSubstitutor
StrLookup
Logger
- name: String
LoggerConfig
- name: String
- parent: LoggerConfig
Appender
- name: String
Layout
Filter
Filter
1
0..*
1..*
0..1

Log4J - Основные аппендеры

- org.apache.log4j.FileAppender - просто записывает логируемые сообщения в файл.
- org.apache.log4j.DailyRollingFileAppender - тоже записывает сообщения в файл, но каждый день создаёт новый файл с таким же именем.
- org.apache.log4j.RollingFileAppender - записывает сообщения в файл и создаёт новые файлы.
- org.apache.log4j.net.SMTPAppender - посылает сообщения по электронной почте.

Log4J - Дополнительные аппендеры

- org.apache.log4j.AsyncAppender - вспомогательный аппендер. Позволяет записывать сообщения асинхронно. Использует буфер для увеличения производительности.
- org.apache.log4j.varia.NullAppender - просто выбрасывает все сообщения (записывает в Null).
- org.apache.log4j.jdbc.JDBCAppender записывает сообщения в БД через jdbc. Уже буферизирован. Как написано в javadoc этот аппендер могут полностью

Log4J - Форматтеры

org.apache.log4j.SimpleLayout - наиболее простой и без всяких настроек. Выводит приоритет, знак "-" и логируемое сообщение.

org.apache.log4j.PatternLayout - гибкий форматер, выводящий строковые сообщения. Результирующая строка в логе получается форматированием LoggingEvent , используя параметр ConversionPattern.

Log4J - Паттерны для PatternLayout

%d{ABSOLUTE}
Выводит время. В скобках можно указать формат вывода в формате SimpleDateFormat.

%5p
Выводит уровень лога (ERROR, DEBUG, INFO и пр.), цифра 5 означает, что всегда использовать 5 символов

%t
Выводит имя потока, который вывел сообщение

%c{1}
Категория, в скобках указывается сколько уровней выдавать.

%M
Имя метода, в котором произошёл вызов записи в лог

%L
Номер строки, в которой произошёл вызов записи в лог

%m
Сообщение, которое передали в лог

%n
Перевод строки

Log4J

```
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.6.1</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.6.1</version>
</dependency>
```

Log4J Конфигурирование

Конфигурирование log4j основано на файле log4j.xml.

Log4J Конфигурирование

Вывода лога в консоль

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="debug" name="baeldung" packages="">
    <Appenders>
        <Console name="stdout" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss} %p %m%n"/>
        </Console>
    </Appenders>
    <Loggers>
        <Root level="error">
            <AppenderRef ref="STDOUT"/>
        </Root>
    </Loggers>
</Configuration>
```

Log4J Конфигурирование

Вывод лога в файл

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="debug" name="baeldung" packages="">
    <Appenders>
        <File name="fout" fileName="baeldung.log" append="true">
            <PatternLayout>
                <Pattern>%d{yyyy-MM-dd HH:mm:ss} %-5p %m%nw</Pattern>
            </PatternLayout>
        </File>
    </Appenders>
    <Loggers>
        <Root level="error">
            <AppenderRef ref="stdout"/>
            <AppenderRef ref=fout""/>
        </Root>
    </Loggers>
</Configuration>
```

Log4J

```
import org.apache.logging.log4j.Logger;
import org.apache.logging.log4j.LogManager;

public class Log4jExample {

    private static Logger logger = LogManager.getLogger(Log4jExample.class);

    public static void main(String[] args) {
        logger.debug("Debug log message");
        logger.info("Info log message");
        logger.error("Error log message");
    }
}
```

SLF4J (Simple Logging Facade for Java)

- SLF4J является более продвинутым, нежели Commons Logging – набор поддерживаемых им возможностей конечных фреймворков шире.
- SLF4J поддерживает (абстрагирует) большее количество фреймворков логирования
- SLF4J является приемником для java.util.logging, Log4J, Commons Logging – т.е. его можно подключить "под" эти фреймворки.
- Нет проблем с загрузчиком классов

Уровни логирования

slf4j	Log4j	Log4j2	Logback	java.util.logging
FATAL	FATAL	FATAL		
ERROR	ERROR	ERROR	ERROR	SEVERE
WARN	WARN	WARN	WARN	WARNING
INFO	INFO	INFO	INFO	INFO
				CONFIG
DEBUG	DEBUG	DEBUG	DEBUG	FINE
				FINER
TRACE	TRACE	TRACE	TRACE	FINEST

Вопросы для самоконтроля

1. Для чего нужен лог файл?
2. Какие основные библиотеки логгирования вы знаете?
3. Зачем нужно логгирование программы?
4. Нужно ли логгировать исключения?